Calibrating Structured Output Predictors for Natural Language Processing

Abhyuday Jagannatha[1], **Hong Yu**[1,2]
[1]College of Information and Computer Sciences,University of Massachusetts Amherst
[2]Department of Computer Science,University of Massachusetts Lowell
{abhyuday, hongyu}@cs.umass.edu

Abstract

We address the problem of calibrating prediction confidence for output entities of interest in natural language processing (NLP) applications. It is important that NLP applications such as named entity recognition and question answering produce calibrated confidence scores for their predictions, especially if the applications are to be deployed in a safety-critical domain such as healthcare. However, the output space of such structured prediction models is often too large to adapt binary or multi-class calibration methods directly. In this study, we propose a general calibration scheme for output entities of interest in neural network based structured prediction models. Our proposed method can be used with any binary class calibration scheme and a neural network model. Additionally, we show that our calibration method can also be used as an uncertainty-aware, entity-specific decoding step to improve the performance of the underlying model at no additional training cost or data requirements. We show that our method outperforms current calibration techniques for named-entity-recognition, part-of-speech and question answering. We also improve our model's performance from our decoding step across several tasks and benchmark datasets. Our method improves the calibration and model performance on out-of-domain test scenarios as well.

1 Introduction

Several modern machine-learning based Natural Language Processing (NLP) systems can provide a confidence score with their output predictions. This score can be used as a measure of predictor confidence. A well-calibrated confidence score is a probability measure that is closely correlated with the likelihood of model output's correctness. As a result, NLP systems with calibrated confidence can predict when their predictions are likely to be incorrect and therefore, should not be trusted. This property is necessary for the responsible deployment of NLP systems in safety-critical domains such as healthcare and finance. Calibration of predictors is a well-studied problem in machine learning (Guo et al., 2017; Platt, 2000); however, widely used methods in this domain are often defined as binary or multi-class problems(Naeini et al., 2015; Nguyen and O'Connor, 2015). The structured output schemes of NLP tasks such as information extraction (IE) (Sang and De Meulder, 2003) and extractive question answering (Rajpurkar et al., 2018) have an output space that is often too large for standard multi-class calibration schemes.

Formally, we study NLP models that provide conditional probabilities $p_\theta(y|x)$ for a structured output y given input x. The output can be a label sequence in case of part-of-speech (POS) or named entity recognition (NER) tasks, or a span prediction in case of extractive question answering (QA) tasks, or a relation prediction in case of relation extraction task. $p_\theta(y|x)$ can be used as a score of the model's confidence in its prediction. However, $p_\theta(y|x)$ is often a poor estimate of model confidence for the output y. The output space of the model in sequence-labelling tasks is often large, and therefore $p_\theta(y|x)$ for any output instance y will be small. For instance, in a sequence labelling task with C number of classes and a sequence length of L, the possible events in output space will be of the order of C^L. Additionally, recent efforts (Guo et al., 2017; Nguyen and O'Connor, 2015; Dong et al., 2018; Kumar and Sarawagi, 2019) at calibrating machine learning models have shown that they are poorly calibrated. Empirical results from Guo et al. (2017) show that techniques used in deep neural networks such as dropout and their large architecture size can negatively affect the calibration of their outputs in binary and multi-class classification tasks.

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 2078–2092
July 5 - 10, 2020. ©2020 Association for Computational Linguistics

Parallelly, large neural network architectures based on contextual embeddings (Devlin et al., 2018; Peters et al., 2018) have shown state-of-the-art performance across several NLP tasks (Andrew and Gao, 2007; Wang et al., 2019) . They are being rapidly adopted for information extraction and other NLP tasks in safety-critical applications (Zhu et al., 2018; Sarabadani, 2019; Li et al., 2019; Lee et al., 2019). Studying the miss-calibration in such models and efficiently calibrating them is imperative for their safe deployment in the real world.

In this study, we demonstrate that neural network models show high calibration errors for NLP tasks such as POS, NER and QA. We extend the work by Kuleshov and Liang (2015) to define well-calibrated forecasters for output entities of interest in structured prediction of NLP tasks. We provide a novel calibration method that applies to a wide variety of NLP tasks and can be used to produce model confidences for specific output entities instead of the complete label sequence prediction. We provide a general scheme for designing manageable and relevant output spaces for such problems. We show that our methods lead to improved calibration performance on a variety of benchmark NLP datasets. Our method also leads to improved out-of-domain calibration performance as compared to the baseline, suggesting that our calibration methods can generalize well.

Lastly, we propose a procedure to use our calibrated confidence scores to re-score the predictions in our defined output event space. This procedure can be interpreted as a scheme to combine model uncertainty scores and entity-specific features with decoding methods like Viterbi. We show that this re-scoring leads to consistent improvement in model performance across several tasks at no additional training or data requirements.

2 Calibration framework for Structured Prediction NLP models

2.1 Background

Structured Prediction refers to the task of predicting a structured output $y = [y_1, y_2, ...y_L]$ for an input x. In NLP, a wide array of tasks including parsing, information extraction, and extractive question answering fall within this category. Recent approaches towards solving such tasks are commonly based on neural networks that are trained by minimizing the following objective :

$$\mathcal{L}(\theta|\mathcal{D}) = -\sum_{i=0}^{|\mathcal{D}|} log(p_\theta(y^{(i)}|x^{(i)})) + R(\theta) \quad (1)$$

where θ is the parameter vector of the neural network and R is the regularization penalty and $\mathcal{D}$ is the dataset $\{(y^{(i)}, x^{(i)})\}_{i=0}^{|\mathcal{D}|}$. The trained model p_θ can then be used to produce the output $\hat{y} = \text{argmax}_{y\in\mathcal{Y}} \, p_\theta(y|x)$. Here, the corresponding model probability $p_\theta(\hat{y}|x)$ is the uncalibrated confidence score.

In binary class classification, the output space $\mathcal{Y}$ is $[0, 1]$. The confidence score for such classifiers can then be calibrated by training a forecaster $F_y : [0, 1] \to [0, 1]$ which takes in the model confidence $F_y(P_\theta(y|x))$ to produce a recalibrated score (Platt, 2000). A widely used method for binary class calibration is Platt scaling where F_y is a logistic regression model. Similar methods have also been defined for multi-class classification (Guo et al., 2017). However, extending this to structured prediction in NLP settings is non-trivial since the output space $|\mathcal{Y}|$ is often too large for us to calibrate the output probabilities of all events.

2.2 Related Work

Calibration methods for binary/multi class classification has been widely studied in related literature (Bröcker, 2009; Guo et al., 2017). Recent efforts at confidence modeling for NLP has focused on several tasks like co-reference, (Nguyen and O'Connor, 2015), semantic parsing (Dong et al., 2018) and neural machine translation (Kumar and Sarawagi, 2019).

2.3 Calibration in Structured Prediction

In this section, we define the calibration framework by Kuleshov and Liang (2015) in the context of structured prediction problems in NLP. The model p_θ denotes the neural network that produces an conditional probability $p_\theta(y|x)$ given an (x, y) tuple. In a multi/binary class setting, a function F_y is used to map the output $p_\theta(y|x)$ to a calibrated confidence score for all $y \in \mathcal{Y}$. In a structured prediction setting, since the cardinality of $\mathcal{Y}$ is usually large, we instead focus on the event of interest set $\mathcal{I}(x)$. $\mathcal{I}(x)$ contains events of interest E that are defined using the output events relevant to the deployment requirements of a model. The event E is a subset of $\mathcal{Y}$. There can be several different schemes to define $\mathcal{I}(x)$. In later sections, we discuss related work on calibration that can be understood as applications of different $\mathcal{I}(x)$ schemes. In this work, we define a general framework for constructing $\mathcal{I}(x)$ for NLP tasks which allows us to maximize calibration performance on output entities of interest.

We define $F_y(E, x, p_\theta)$ to be a function, that takes the event E, the input feature x and p_θ to produce a confidence score between $[0, 1]$. We refer to this calibration function as the forecaster and use $F_y(E, x)$ as a shorthand since it is implicit that F_y depends on outputs of p_θ. We would like to find the forecaster that minimizes the discrepancy between $F_y(E, x)$ and $\mathbb{P}(y \in E|x)$ for (x, y) sampled from $\mathbb{P}(x, y)$ and E uniformly sampled from $\mathcal{I}(x)$.

A commonly used methodology for constructing a forecaster for p_θ is to train it on a held-out dataset D_{dev}. A forecaster for a binary classifier is perfectly calibrated if

$$\mathbb{P}(y = 1|F_y(x) = p) = p. \qquad (2)$$

It is trained on samples from $\{(x, \mathbb{I}(y = 1) : (x, y) \in D_{dev}\}$. For our forecaster based on $\mathcal{I}(x)$, perfect calibration would imply that

$$\mathbb{P}(y \in E|F_y(x, E) = p) = p. \qquad (3)$$

The training data samples for our forecaster are $\{(x, \mathbb{I}(y \in E) : E \in \mathcal{I}(x), (x, y) \in D_{dev}\}$.

2.4 Construction of Event of Interest set $\mathcal{I}(x)$

The main contributions of this paper stem from our proposed schemes for constructing the aformentioned $\mathcal{I}(x)$ sets for NLP applications.

Entities of Interest : In the interest of brevity, let us define "Entities of interest" $\phi(x)$ as the set of all entity predictions that can be queried from p_θ for a sample x. For instance, in the case of answer span prediction for QA, the $\phi(x)$ may contain the MAP prediction of the best answer span (answer start and end indexes). In a parsing or sequence labeling task, $\phi(x)$ may contain the top-k label sequences obtained from viterbi decoding. In a relation or named-entity extraction task, $\phi(x)$ contains the relation or named entity span predictions respectively. Each entity s in $\phi(x)$ corresponds to a event set E that is defined by all outputs in $\mathcal{Y}$ that contain the entity s. $\mathcal{I}(x)$ contains set E for all entities in $\phi(x)$.

Positive Entities and Events : We are interested in providing a calibrated probability for $y \in E$ corresponding to an s for all s in $\phi(x)$. Here y is the correct label sequence for the input x. If y lies in the set E for an entity s, we refer to s as a positive entity and the event as a positive event. In the example of named entity recognition, s may refer to a predicted entity span, E refers to all possible sequences in $\mathcal{Y}$ that contain the predicted span. The corresponding event is positive if the correct label sequence y contains the span prediction s.

Schemes for construction of $\mathcal{I}(x)$: While constructing the set $\phi(x)$ we should ensure that it is limited to a relatively small number of output entities, while still covering as many positive events in $\mathcal{I}(x)$ as possible. To explain this consideration, let us take the example of a parsing task such as syntax or semantic parsing. Two possible schemes for defining $\mathcal{I}(x)$ are :

1. Scheme 1: $\phi(x)$ contains the MAP label sequence prediction. $\mathcal{I}(x)$ contains the event corresponding to whether the label sequence $y' = \mathrm{argmax}_y\, p_\theta(y|x)$ is correct.

2. Scheme 2: $\phi(x)$ contains all possible label sequences. $\mathcal{I}(x)$ contains a event corresponding to whether the label sequence y' is correct, for all $y' \in \mathcal{Y}$

Calibration of model confidence by Dong et al. (2018) can be viewed as Scheme 1, where the entity of interest is the MAP label sequence prediction. Whereas, using Platt Scaling in a one-vs-all setting for multi-class classification (Guo et al., 2017) can be seen as an implementation of Scheme 2 where the entity of interest is the presence of class label. As discussed in previous sections, Scheme 2 is too computationally expensive for our purposes due to large value of $|\mathcal{Y}|$. Scheme 1 is computationally cheaper, but it has lower coverage of positive events. For instance, a sequence labelling model with a 60% accuracy at sentence level means that only 60 % of positive events are covered by the set corresponding to $\mathrm{argmax}_y\, p_\theta(y|x)$ predictions. In other words, only 60 % of the correct outputs of model p_θ will be used for constructing the forecaster. This can limit the positive events in $\mathcal{I}(x)$. Including the top-k predictions in $\phi(x)$ may increase the coverage of positive events and therefore increase the positive training data for the forecaster. The optimum choice of k involves a trade-off. A larger value of k implies broader coverage of positive events and more positive training data for the forecaster training. However, it may also lead to

Calibration	BERT	BERT+CRF	DistilBERT
Platt	15.90±.03	15.56±.23	12.30±.13
Calibrated Mean	2.55±.34	**2.31±.35**	**2.02±.16**
+Var	**2.11±.32**	2.55±.32	2.73±.40
Platt+top2	11.4±.07	14.21±.16	11.03±.31
Calibrated Mean+top2	2.94± .29	4.82±.15	3.61±.17
+Var+top2	2.17±.35	4.26±.10	2.43±.16
+Rank+top2	2.43±.30	2.43±.45	2.21±.09
+Rank+Var+top2	**1.81±.12**	**2.29±.27**	**1.97±.14**
Platt+top3	17.46±.13	18.11±.16	12.84±.37
+Rank+Var+top3	3.18±.12	3.71±.25	2.05±.06

Table 1: ECE percentages on Penn Treebank for different models and calibration methods. The results are for top-1 MAP predictions on the test data. ECE standard deviation is estimated by repeating the experiments for 5 repetitions. ECE for uncalibrated BERT, BERT+CRF model and DistilBERT is 35.11%, 33.72% and 28.06% respectively. *heuristic-k* is 2 for all +Rank+Var+topk forecasters. Full feature model +Rank+Var+topk, $k = 3$ is also provided for completeness.

an unbalanced training dataset that is skewed in favour of negative training examples.

Task specific details about $\phi(x)$ are provided in the later sections. For the purposes of this paper, top-k refers to the top k MAP sequence predictions, also referred to as $argmax(k)$.

2.5 Forecaster Construction

Here we provide a summary of the steps involved in Forecaster construction. Remaining details are in the Appendix. We train the neural network model p_θ on the training data split for a task and use the validation data for monitoring the loss and early stopping. After the training is complete, this validation data is re-purposed to create the forecaster training data. We use an MC-Dropout(Gal and Ghahramani, 2016) average of (n=10) samples to get a low variance estimate of logit outputs from the neural networks. This average is fed into the decoding step of the model p_θ to obtain top-k label sequence predictions. We then collect the relevant entities in $\phi(x)$, along with the $\mathbb{I}(y \in E)$ labels to form the training data for the forecaster. We use gradient boosted decision trees (Friedman, 2001) as our region-based (Dong et al., 2018; Kuleshov and Liang, 2015) forecaster model.

Choice of the hyperparameter k: We limit our choice of k to $\{2, 3\}$. We train our forecasters on training data constructed through top-2 and top-3 extraction each. These two models are then evaluated on top-1 extraction training data, and the best value of k is used for evaluation on test. This heuristic for k selection is based on the fact that the top-1 training data for a *good* predictor p_θ, is a positive-event rich dataset. Therefore, this dataset can be used to reject a larger k if it leads to reduced performance on positive events. We refer to the value of k obtained from this heuristic as as *heuristic-k*.

2.6 Feature Construction for Calibration

We use three categories of features as inputs to our forecaster.

Model and Model Uncertainty based features contain the mean probability obtained by averaging over the marginal probability of the "entity of interest" obtained from 10 MC-dropout samples of p_θ. Average of marginal probabilities acts as a reduced variance estimate of un-calibrated model confidence. Our experiments use the pre-trained contextual word embedding architectures as the backbone networks. We obtain MC-Dropout samples by enabling dropout sampling for all dropout layers of the networks. We also provide 10^{th} and 90^{th} percentile values from the MC-Dropout samples, to provide model uncertainty information to the forecaster. Since our forecaster training data contains entity predictions from top-k MAP predictions, we also include the rank k as a feature. We refer to these two features as "Var" and "Rank" in our models.

Entity of interest based features contain the length of the entity span if the output task is named entity. We only use this feature in the NER experiments and refer to it as "ln".

Data Uncertainty based features: Dong et al. (2018) propose the use of language modelling (LM)

Calibration	BERT	BERT+CRF	DistilBERT
Baseline	60.30±.12	62.31±.11	60.17±.08
+Rank+Var+top2	60.30±.23	62.31±.09	60.13±.11
+Rank+Var+top3	59.84±.16	61.06±.14	58.95±.08

Table 2: Micro-avg f-score for POS datasets using the baseline and our best proposed calibration method. The confidence score from the calibration method is used to re-rank the events $E \in \mathcal{I}(s)$ and the top selection is chosen. Standard deviation is estimated by repeating the experiments for 5 repetitions. Baseline refers to MC-dropout averaged (sample-size=10) output from the model p_θ. *heuristic-k* is 2 for +Rank+Var+topk forecasters.

and OOV-word-based features as a proxy for data uncertainty estimation. The use of word-pieces and large pre-training corpora in contextual word embedding models like BERT may affect the efficacy of LM based features. Nevertheless, we use LM perplexity (referred to as "lm") in the QA task to investigate its effectiveness as an indicator of the distributional shift in data. Essentially, our analysis focuses on LM perplexity as a proxy for *distributional* uncertainty (Malinin and Gales, 2018) in our out-of-domain experiments. The use of word-pieces in models like BERT reduces the negative effect of OOV words on model prediction. Therefore, we do not include OOV features in our experiments.

3 Experiments and Results

We use BERT-base (Devlin et al., 2018) and distilBERT (Sanh et al., 2019) network architecture for our experiments. Validation split for each dataset was used for early stopping BERT fine-tuning and as training data for forecaster training. POS and NER experiments are evaluated on Penn Treebank and CoNLL 2003 (Sang and De Meulder, 2003), MADE 1.0 (Jagannatha et al., 2019) respectively. QA experiments are evaluated on SQuAD1.1 (Rajpurkar et al., 2018) and EMRQA (Pampari et al., 2018) corpus. We also investigate the performance of our forecasters on an out-of-domain QA corpus constructed by applying EMRQA QA data generation scheme (Pampari et al., 2018) on the MADE 1.0 named entity and relations corpus. Details for these datasets are provided in their relevant sections.

We use the expected calibration error (ECE) metric defined by Naeini et al. (2015) with $N = 20$ bins (Guo et al., 2017) to evaluate the calibration of our models. ECE is defined as an estimate of the expected difference between the model confidence and accuracy. ECE has been used in several related works (Guo et al., 2017; Maddox et al., 2019; Kumar et al., 2018; Vaicenavicius et al., 2019) to estimate model calibration. We use Platt scaling as the baseline calibration model. It uses the length-normalized probability averaged across 10 MC-Dropout samples as the input. The lower variance and length invariance of this input feature make Platt Scaling a strong baseline. We also use a "Calibrated Mean" baseline using Gradient Boosted Decision Trees as our estimator with the same input feature as Platt.

3.1 Calibration for Part-of-Speech Tagging

Part-of-speech (POS) is a sequence labelling task where the input is a text sentence, and the output is a sequence of syntactic tags. We evaluate our method on the Penn Treebank dataset (Marcus et al., 1994). We can define either the token prediction or the complete sequence prediction as the entity of interest. Since using a token level entity of interest effectively reduces the calibration problem to that of calibrating a multi-class classifier, we instead study the case where the predicted label sequence of the entire sentence forms the entity of interest set. The event of interest set is defined by the events $y = \text{MAP}_k(x)$ which denote whether each top-k sentence level MAP prediction is correct. We use three choice of p_θ models, namely BERT, BERT-CRF and distilBERT. We use model uncertainty and rank based features for our POS experiments.

Table 1 shows the ECE values for our baseline, proposed and ablated models. The value of *heuristic-k* is 2 for all +Rank+Var+topk forecasters across all PTB models. "top$k$" in Table 1 refers to forecasters trained with additional top-k predictions. Our methods outperform both baselines by a large margin. Both "Rank" and "Var" features help in improving model calibration. Inclusion of top-2 prediction sequences also improve the calibration performance significantly. Table 1 also shows the performance of our full feature model "+Rank+Var+top$k$" for the sub-optimal value of

Calibration	CoNLL (BERT)	MADE 1.0 (bioBERT)
Platt	**2.00**±**.12**	4.00±.07
Calibrated Mean	2.29±.33	3.07±.18
+Var	2.43±.36	3.05±.17
+Var+ln	2.24±.14	**2.92**±**.24**
Platt+top3	16.64±.48	2.14±.18
Calibrated Mean+top3	17.06±.50	2.22±.31
+Var+top3	17.10±.24	2.17±.39
+Rank+Var+top3	2.01±.33	2.34±.15
+Rank+Var+ln+top3	**1.91**±**.29**	**2.12**±**.24**

Table 3: ECE percentages for the two named entity datasets and calibration methods. The results are for all predicted named entity spans in top-1 MAP predictions on the test data. ECE standard deviation is estimated by repeating the experiments for 5 repetitions. ECE for uncalibrated span marginals from BERT model is 3.68% and 5.59% for CoNLL and MADE 1.0 datasets. *heuristic-k* is 3 for all +Rank+Var+top3 forecasters.

Calibration	CoNLL (BERT)	MADE 1.0 (bioBERT)
Baseline	89.45±.08	84.01±.11
+Rank+Var+top3	89.73±.12	84.33±.07
+Rank+Var+ln+top3	**89.78**±**.10**	**84.34**±**.10**

Table 4: Micro-avg f-score for NER datasets and our best proposed calibration method. The confidence score from the calibration method is used to re-rank the events $E \in \mathcal{I}(s)$ and a confidence value of 0.5 is used as a cutoff. Standard deviation is estimated by repeating the experiments for 5 repetitions. Baseline refers to MC-dropout averaged (sample-size=10) output of model p_θ. *heuristic-k* is 3 for all +Rank+Var+top3 forecasters.

$k = 3$. It has lower performance than $k = 2$ across all models. Therefore for the subsequent experimental sections, we only report top$k$ calibration performance using the *heuristic-k* value only.

We use the confidence predictions of our full-feature model +Rank+Var+top$k$ to re-rank the top-k predictions in the test set. Table 2 shows the sentence-level (entity of interest) accuracy for our re-ranked top prediction and the original model prediction.

3.2 Calibration for Named Entities

For Named Entity (NE) Recognition experiments, we use two NE annotated datasets, namely CoNLL 2003 and MADE 1.0. CoNLL 2003 consists of documents from the Reuters corpus annotated with named entities such as Person, Location etc. MADE 1.0 dataset is composed of electronic health records annotated with clinical named entities such as Medication, Indication and Adverse effects.

The entity of interest for NER is the named entity span prediction. We define $\phi(x)$ as predicted entity spans in argmax(k) label sequences predictions for x. We use BERT-base with token-level softmax output and marginal likelihood based training. The model uncertainty estimates for "Var" feature are computed by estimating the variance of length normalized MC-dropout samples of span marginals. Due to the similar trends in behavior of BERT and BERT+CRF model in POS experiments, we only use BERT model for NER. However, the span marginal computation can be easily extended to linear-chain CRF models. We also use the length of the predicted named entity as the feature "ln" in this experiment. Complete details about forecaster and baselines are in the Appendix. Value of *heuristic-k* is 3 for all +Rank+Var+topk forecasters. We show ablation and baseline results for $k = 3$ only. However, no other forecasters for any $k \in \{2, 3\}$ outperform our best forecasters in Table 3.

We use the confidence predictions of our "+Rank+Var+top3" models to re-score the confidence predictions for all spans predicted in top-3 MAP predictions for samples in the test set. A threshold of 0.5 was used to remove span predictions with low confidence scores. Table 4 shows the Named Entity level (entity of interest) Micro-F score for our re-ranked top prediction and the original model prediction. We see that re-ranked predictions from our models consistently improve the model f-score.

3.3 Calibration for QA Models

We use three datasets for evaluation of our calibration methods on the QA task. Our QA tasks are modeled as extractive QA methods with a single span answer predictions. We use three datasets to construct experiments for QA calibration. SQuAD1.1 and EMRQA (Pampari et al., 2018) are open-domain and clinical-domain QA datasets, respectively. We process the EMRQA dataset by restricting the passage length and removing unanswerable questions. We also design an out-of-domain evaluation of calibration using clinical QA datasets. We follow the guidelines from Pampari et al. (2018) to create a QA dataset

Calibration	**SQuAD1.1** (BERT)	**EMRQA** (bioBERT)	**MADE 1.0** (bioBERT)	**MADE 1.0(OOD)** (bioBERT)
Platt	3.69±.16	5.07±.37	3.64±.17	15.20±.16
Calibrated Mean	2.95±.26	2.28±.18	2.50±.31	13.26±.94
+Var	**2.92**±**.28**	2.74±.15	2.71±.32	**12.41**±**.95**
Platt+top3	7.71±.28	5.42±.25	11.87±.19	16.36±.26
Calibrated Mean+top3	3.52±.35	2.11±.19	9.21±.25	12.11±.24
+Var+top3	3.56±.29	2.20±.20	9.26±.27	**11.67**±**.27**
+Var+lm+top3	3.54±.21	2.12±.19	6.07±.26	12.42±.32
+Rank+Var+top3	**2.47**±**.18**	**1.98**±**.10**	1.77±.23	12.69±.20
+Rank+Var+lm+top3	2.79±.32	2.24±.29	**1.66**±**.27**	12.60±.28

Table 5: ECE percentages for QA tasks SQuAD1.1, EMRQA and MADE 1.0. MADE 1.0(OOD) refers to the out-of-domain evaluation of a QA model that is trained and calibrated on EMRQA training and validation splits. The results are for top-1 MAP predictions on the test data. ECE standard deviation is estimated by repeating the experiments for 5 repetitions. BERT model's uncalibrated ECE for SQuAD1.1, EMRQA, MADE 1.0 and MADE 1.0(OOD) are 6.24% 6.10%, 20.10% and 18.70% respectively. *heuristic-k* is 3 for all +Rank+Var+top$k$ forecasters.

Calibration	**SQuAD1.1** (BERT)	**EMRQA** (bioBERT)	**MADE 1.0** (bioBERT)	**MADE 1.0(OOD)** (bioBERT)
Baseline	79.79±.08	70.97±.14	66.21±.18	31.62±.12
+Rank+Var+top3	**80.04**±**.11**	71.34±.22	**66.33**±**.12**	31.99±.11
+Rank+Var+lm+top3	**80.03**±**.15**	**71.37**±**.26**	**66.33**±**.15**	**32.02**±**.09**

Table 6: Table shows change in Exact Match Accuracy for QA datasets and our best proposed calibration method. The confidence score from the calibration method is used to re-rank the events $E \in \mathcal{I}(s)$. Standard deviation is estimated by repeating the experiments for 5 repetitions. Baseline refers to MC-dropout averaged (sample-size=10) output of model p_θ. *heuristic-k* is 3 for all +Rank+Var+top$k$ forecasters.

from MADE 1.0 (Jagannatha et al., 2019). This allows us to have two QA datasets with common question forms, but different text distributions. In this experimental setup we can mimic the evaluation of calibration methods in a real-world scenario, where the task specifications may remain the same but the underlying text source changes. Details about dataset pre-processing and construction are provided in the Appendix.

The entity of interest for QA is the top-k answer span predictions. We use the "lm" perplexity as a feature in this experiment to analyze its behaviour in out-of-domain evaluations. We use a 2 layer unidirectional LSTM to train a next word language model on the EMRQA passages. This language model is then used to compute the perplexity of a sentence for the "lm" input feature to the forecaster. We use the same baselines as the previous two tasks.

Based on Table 5, our methods outperform the baselines by a large margin in both in-domain and out-of-domain experiments. Value of *heuristic-k* is 3 for all +Rank+Var+topk forecasters. We show ablation and baseline results for $k = 3$ only. However, no other forecasters for any $k \in \{2, 3\}$ outperform our best forecasters in Table 5 . Our models are evaluated on SQuAD1.1 dev set, and test sets from EMRQA and MADE 1.0. They show consistent improvements in ECE and Exact Match Accuracy.

4 Discussion

Our proposed methods outperform the baselines in most tasks and are almost as competitive in others.

Features and top-k samples: The inclusion of top-k features improve the performance in almost all tasks when the rank of the prediction is included. We see large increases in calibration error when the top-k prediction samples are included in forecaster training without including the rank information in tasks such as CoNLL NER and MADE 1.0 QA. This may be because the $k = 1, 2, 3$ predictions

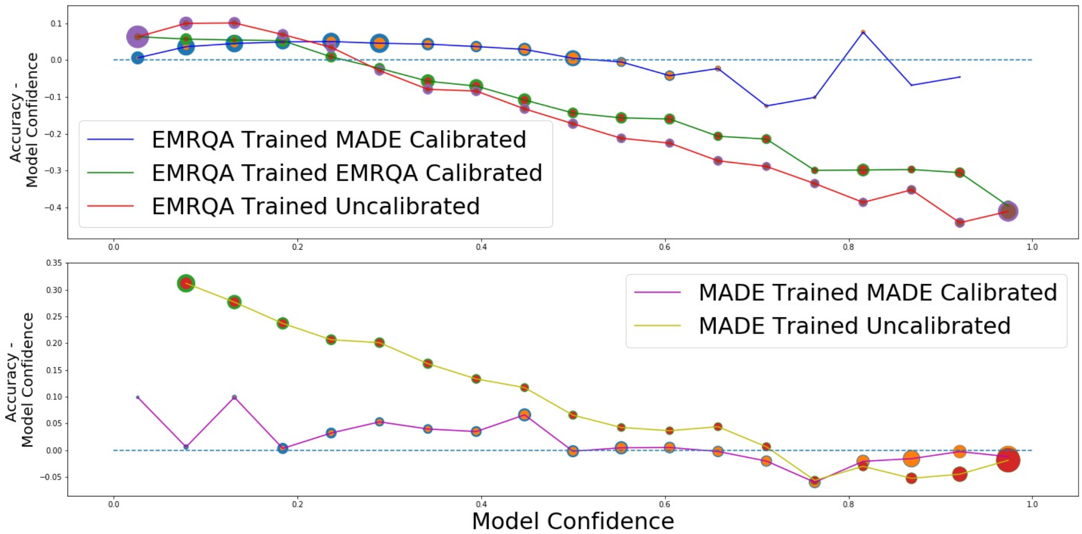

Figure 1: Modified reliability plots (*Accuracy - Confidence* vs *Confidence*) on MADE 1.0 QA test. The dotted horizontal line represents perfect calibration. Scatter point diameter denotes bin size. The inner diameter of the scatter point denotes the number of positive events in that bin.

may have similar model confidence and uncertainty values. Therefore a more discriminative signal such as rank is needed to prioritize them. For instance, the difference between probabilities of $k = 1$ and $k = 2$ MAP predictions for POS tagging may differ by only one or two tokens. In a sentence of length 10 or more, this difference in probability when normalized by length would account to very small shifts in the overall model confidence score. Therefore an additional input of rank k leads to a substantial gain in performance for all models in POS.

Our task-agnostic scheme of "Rank+Var+topk" based forecasters consistently outperform or stay competitive to other forecasting methods. However, results from task-specific features such as "lm" and "len" show that use of task-specific features can further reduce the calibration error. Our domain shift experimental setup has the same set of questions in both in-domain and out-of-domain datasets. Only the data distribution for the answer passage is different. However, we do not observe an improvement in out-of-domain performance by using "lm" feature. A more detailed analysis of task-specific features in QA with both data and question shifts is required. We leave further investigations of such schemes as our future work.

"By the close Yorkshire had turned that into a 37-run advantage but off-spinner Such had scuttled their hopes , taking four for 24 in 48 balls and leaving them hanging on 119 for five and praying for rain."

Entity	Rank	Mean Prob	Calibrated Confidence
off-spinner Such	Rank 1	90.08± 8.65	86.32
Such	Rank 2	89.02 ± 2.07	86.61

Figure 2: An example of named entity span from CoNLL dataset. Rank is k^{th} rank from top-k MAP inference (Viterbi decoding). Mean Prob and Std is the mean and standard deviation of length-normalized probabilities (geometric mean of marginal probabilities for each token in the span). Calibrated confidence is the output of *Rank+Var+ln+top3*.

Choice of k is important : The optimal choice of k seems to be strongly dependent on the inherent properties of the tasks and its output event set. In all our experiments, for a specific task all +Rank+Var+top$k$ forecasters exhibit consistent behaviours with respect to the choice of k. In POS experiments, *heuristic-*k $= 2$. In all other tasks, *heuristic-*k $= 3$. Our *heuristic-*k models are the best performing models, suggesting that the heuristic described in Section 2.5 may generalize to other tasks as well.

Re-scoring : We show that using our forecaster confidence to re-rank the entities of interest leads to a modest boost in model performance for the NER and QA tasks. In POS no appreciable gain or drop in performance was observed for $k = 2$. We believe this may be due to the already high token level accuracy (above 97%) on Penn Treebank data. Nevertheless, this suggests that our re-scoring does not lead to a degradation in model performance in cases where it is not effective.

Our forecaster re-scores the top-k entity confidence scores based on model uncertainty score and entity-level features such as entity lengths. Intuitively, we want to prioritize predictions that have low uncertainty over high uncertainty predictions, if their uncalibrated confidence scores are similar. We provide an example of such re-ranking in Figure 2. It shows a named entity span predictions for the correct span "Such". The model p_θ produces two entity predictions "off-spinner Such" and "Such". The un-calibrated confidence score of "off-spinner Such" is higher than "Such", but the variance of its prediction is higher as well. Therefore the *+Rank+Var+ln+top3* re-ranks the second (and correct) prediction higher. It is important to note here that the variance of "off-spinner Such" may be higher just because it involves two token predictions as compared to only one token prediction in "Such". This along with the "ln" feature in *+Rank+Var+ln+top3* may mean that the forecaster is also using length information along with uncertainty to make this prediction. However, we see similar improvements in QA tasks, where the "ln" feature is not used, and all entity predictions involve two predictions (span start and end index predictions). These results suggest that use of uncertainty features are useful in both calibration and re-ranking of predicted structured output entities.

Out-of-domain Performance : Our experiments testing the performance of calibrated QA systems on out-of-domain data suggest that our methods result in improved calibration on unseen data as well. Additionally, our methods also lead to an improvement in system accuracy on out-of-domain data, suggesting that the mapping learned by the forecaster model is not specific to a dataset. However, there is still a large gap between the calibration error for within domain and out-of-domain testing. This can be seen in the reliability plot shown in Figure 1. The number of samples in each bin are denoted by the radius of the scatter point. The calibrated models shown in the figure corresponds to "+Rank+Var+lm+top3' forecaster calibrated using both in-domain and out-of-domain validation datasets for forecaster training. We see that out-of-domain forecasters are over-confident and this behaviour is not mitigated by using data-uncertainty aware features like "lm". This is likely due to a shift in model's prediction error when applied to a new dataset. Re-calibration of the forecaster using a validation set from the out-of-domain data seems to bridge the gap. However, we can see that the *sharpness* (Kuleshov and Liang, 2015) of *out-of-domain trained, in-domain calibrated* model is much lower than that of *in-domain trained, in-domain calibrated* one. Additionally, a validation dataset is often not available in the real world. Mitigating the loss in calibration and sharpness induced by out-of-domain evaluation is an important avenue for future research.

Uncertainty Estimation : We use MC-Dropout as a model (*epistemic*) uncertainty estimation method in our experiments. However, our method is not specific to MC-Dropout, and is compatible with any method that can provide a predictive distribution over token level outputs. As a result any bayesian or ensemble based uncertainity estimation method (Welling and Teh, 2011; Lakshminarayanan et al., 2017; Ritter et al., 2018) can be used with our scheme. In this work, we do not investigate the use of *aleatoric* uncertainty for calibration. Our use of language model features is aimed at accounting for *distributional* uncertainty instead of *aleatoric* uncertainty (Gal, 2016; Malinin and Gales, 2018). Investigating the use of different types of uncertainty for calibration remains as our future work.

5 Conclusion

We show a new calibration and confidence based re-scoring scheme for structured output entities in NLP. We show that our calibration methods outperform competitive baselines on several NLP tasks. Our task-agnostic methods can provide calibrated model outputs of specific entities instead of the entire label sequence prediction. We also show that our calibration method can provide improvements to the trained model's accuracy at no additional training or data cost. Our method is compatible with modern NLP architectures like BERT. Lastly, we show that our calibration does not over-fit on in-domain data and is capable of generalizing the calibration to out-of-domain datasets.

Acknowledgement

Research reported in this publication was supported by the National Heart, Lung, and Blood Institute (NHLBI) of the National Institutes of Health under Award Number R01HL125089.

References

Galen Andrew and Jianfeng Gao. 2007. Scalable training of L1-regularized log-linear models. In *Proceedings of the 24th International Conference on Machine Learning*, pages 33–40.

Jochen Bröcker. 2009. Reliability, sufficiency, and the decomposition of proper scores. *Quarterly Journal of the Royal Meteorological Society: A journal of the atmospheric sciences, applied meteorology and physical oceanography*, 135(643):1512–1519.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Li Dong, Chris Quirk, and Mirella Lapata. 2018. Confidence modeling for neural semantic parsing. *arXiv preprint arXiv:1805.04604*.

Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Yarin Gal. 2016. Uncertainty in deep learning. *University of Cambridge*, 1:3.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1321–1330. JMLR. org.

Abhyuday Jagannatha, Feifan Liu, Weisong Liu, and Hong Yu. 2019. Overview of the first natural language processing challenge for extracting medication, indication, and adverse drug events from electronic health record notes (made 1.0). *Drug safety*, 42(1):99–111.

Volodymyr Kuleshov and Percy S Liang. 2015. Calibrated structured prediction. In *Advances in Neural Information Processing Systems*, pages 3474–3482.

Aviral Kumar and Sunita Sarawagi. 2019. Calibration of encoder decoder models for neural machine translation. *arXiv preprint arXiv:1903.00802*.

Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. 2018. Trainable calibration measures for neural networks from kernel mean embeddings. In *International Conference on Machine Learning*, pages 2810–2819.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. 2019. Biobert: pre-trained biomedical language representation model for biomedical text mining. *arXiv preprint arXiv:1901.08746*.

Fei Li, Yonghao Jin, Weisong Liu, Bhanu Pratap Singh Rawat, Pengshan Cai, and Hong Yu. 2019. Fine-tuning bidirectional encoder representations from transformers (bert)–based models on large-scale electronic health record notes: An empirical study. *JMIR Med Inform*, 7(3):e14830.

Wesley Maddox, Timur Garipov, Pavel Izmailov, Dmitry Vetrov, and Andrew Gordon Wilson. 2019. A simple baseline for bayesian uncertainty in deep learning. *arXiv preprint arXiv:1902.02476*.

Andrey Malinin and Mark Gales. 2018. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems*, pages 7047–7058.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: annotating predicate argument structure. In *Proceedings of the workshop on Human Language Technology*, pages 114–119. Association for Computational Linguistics.

Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. 2015. Obtaining well calibrated probabilities using bayesian binning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Cláudio A Naranjo, Usoa Busto, Edward M Sellers, P Sandor, I Ruiz, EA Roberts, E Janecek, C Domecq, and DJ Greenblatt. 1981. A method for estimating the probability of adverse drug reactions. *Clinical Pharmacology & Therapeutics*, 30(2):239–245.

Khanh Nguyen and Brendan O'Connor. 2015. Posterior calibration and exploratory analysis for natural language processing models. *arXiv preprint arXiv:1508.05154*.

Anusri Pampari, Preethi Raghavan, Jennifer Liang, and Jian Peng. 2018. emrqa: A large corpus for question answering on electronic medical records. *arXiv preprint arXiv:1809.00732*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

J Platt. 2000. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. *Advances in Large Margin Classifiers*, pages 61–74.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Hippolyt Ritter, Aleksandar Botev, and David Barber. 2018. A scalable laplace approximation for neural networks. In *6th International Conference on Learning Representations, ICLR 2018-Conference Track Proceedings*, volume 6.

Erik F Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Sarah Sarabadani. 2019. Detection of adverse drug reaction mentions in tweets using elmo. In *Proceedings of the Fourth Social Media Mining for Health Applications (# SMM4H) Workshop & Shared Task*, pages 120–122.

Juozas Vaicenavicius, David Widmann, Carl Andersson, Fredrik Lindsten, Jacob Roll, and Thomas B Schön. 2019. Evaluating model calibration in classification. *arXiv preprint arXiv:1902.06977*.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint arXiv:1905.00537*.

Max Welling and Yee W Teh. 2011. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Henghui Zhu, Ioannis Ch Paschalidis, and Amir Tahmasebi. 2018. Clinical concept extraction with contextual word embedding. *arXiv preprint arXiv:1810.10566*.

A Appendices

A.1 Algorithm Details:

The forecaster construction algorithm is provided in Algorithm 1. The candidate events in Algorithm 1 are obtained by extracting top-k label sequences for every output. The logits obtained from p_θ are averaged over 10 MC-Dropout samples before being fed into the final output layer. We use the validation dataset from the task's original split to train the forecaster. The validation dataset is used to construct both training and validation split for the forecaster. The training split contains all top-k predicted entities. The validation split contains only top-1 predicted entities.

A.2 Evaluation Details

We use the expected calibration error (ECE) score defined by (Naeini et al., 2015) to evaluate our calibration methods. Expected calibration error is a score that estimates the expected absolute difference between model confidence and accuracy. This is calculated by binning the model outputs into N ($N = 20$ for our experiments) bins and then computing the expected calibration error across all bins. It is defined as

$$ECE = \sum_{i=0}^{N} \frac{|B_i|}{n} |acc(B_i) - conf(B_i)|, \quad (4)$$

where N is the number of bins, n is the total number of data samples, B_i is the i^{th} bin. The functions $acc(.)$ and $conf(.)$ calculate the accuracy and model confidence for a bin.

A.3 Implementation Details

We use AllenNLP's wrapper with HuggingFace's Transformers code [1] for our implementation[2]. We use BERT-*base-cased* (Wolf et al., 2019) weights as the initialization for general-domain datasets and *bio*-BERT weights (Lee et al., 2019) as the initialization for clinical datasets. We use cased models for our analysis, since bio-BERT(Lee et al., 2019) uses cased models. A common learning rate of 2e-5 was used for all experiments. We used validation data splits provided by the datasets. In cases where the validation dataset was not provided, such as MADE 1.0, EMRQA or SQuAD1.1, we use 10% of the training data as the validation data. We use a patience of 5 for early stopping the model, with each epoch consisting of 20,000 steps. We use the final evaluation metric instead of negative log likelihood (NLL) to monitor and early stop the training. This is to reduce the mis-calibration of the underlying p_θ model, since Guo et al. (2017) observe that neural nets overfit on NLL. The implementation for each experiment is provided in the following subsections.

[1]https://github.com/huggingface/transformers

[2]The code for forecaster construction is available at https://github.com/abhyudaynj/ StructuredPredictionCalibrationNLP

A.3.1 Part-of-speech experiments

We evaluate our method on the Penn Treebank dataset (Marcus et al., 1994). Our experiment uses the standard training (1-18), validation(19-21) and test (22-24) splits from the WSJ portion of the Penn Treebank dataset. The un-calibrated output of our model for a candidate label sequence is estimated as

$$\hat{p} = \frac{1}{M} \sum_{MC-Dropout} p_\theta(y_1, y_2, ...y_L|x)^{\frac{1}{L}}, \quad (5)$$

where M is the number of dropout samples. The L^{th} root accounts for different sentence lengths. Here L is the length of the sentence. We observe that this kind of normalization improves the calibration of both baselines and proposed models. We do not normalize the probabilities while reporting the ECE of uncalibrated models. We use two choice of p_θ models, namely BERT and BERT+CRF. BERT only model adds a linear layer to the output of BERT network and uses a softmax activation function to produce marginal label probabilities for each token. BERT+CRF uses a CRF layer on top of unary potentials obtained from the BERT network outputs.

We use Platt Scaling (Platt, 2000) as the baseline calibration model. Our Platt scaling model uses the MC-Dropout average of length normalized probability output of the model p_θ as input. The lower variance and length invariance of this input feature make Platt Scaling a very strong baseline. We also use a "Calibrated Mean" baseline using Gradient Boosted Decision Trees as our estimator with the same input feature as Platt.

A.3.2 NER Experiments

For CoNLL dataset, "testa" file was reserved for validation data and "testb" was reserved for test. For MADE 1.0 (Jagannatha et al., 2019), since validation data split was not provided we randomly selected 10% of training data as validation data.

Algorithm 1: Forecaster construction for model p_θ with max rank k_{max}.
Input: Uncalibrated model p_θ , Validation Dataset $\mathbb{D} = \{(x^{(i)}, y^{(i)}\}_{i=0}^{|\mathbb{D}|}$, k_{max}.
Output: Forecaster F_y

Function *Get-Forecaster* (p_θ, $\mathbb{D}$, k_{max})
  for $i \leftarrow 0$ **to** $|\mathbb{D}|$ **do**
    $\mathcal{I}(x^{(i)}) \leftarrow$ Get-Candidate-Events$(p_\theta, x^{(i)}, k_{max})$
    $\mathbb{D}_{train} \leftarrow \{(x^{(i)}, c, E) : c = \mathbb{1}(y^{(i)} \in E) \text{ , } \forall E \in \mathcal{I}(x)\}$
    $\mathcal{I}_{k=1}(x^{(i)}) \leftarrow$ Get-Candidate-Events$(p_\theta, x^{(i)}, 1)$
    $\mathbb{D}_{val} \leftarrow \{(x^{(i)}, c, E) : c = \mathbb{1}(y^{(i)} \in E) \text{ , } \forall E \in \mathcal{I}_{k=1}(x)\}$
  end
  Train Forecasters $F_y^{(k)}$ for $k = \{1, ..., k_{max}\}$ using $\mathbb{D}_{train}$
  $F_y \leftarrow F_y^{(k)}$ with minimum ECE on $\mathbb{D}_{val}$
  return F_y

Function *Get-Candidate-Events* (p_θ,x,k_{max})
  Construct top-k_{max} label sequences using MC-Dropout average of $p_\theta(x)$ logits.
  Extract relevant entity set $\phi(x)$ from top-k_{max} label sequences.
  $\mathcal{I}(x) \leftarrow$ Events corresponding to entities in $\phi(x)$.
  return $\mathcal{I}(x)$;

The length normalized marginal probability for a span starting at i and of length l is estimated as

$$\hat{p} = \frac{1}{M} \sum_{MC-Dropout} p_\theta(y_i, y_2, ...y_{i+l-1}|x)^{\frac{1}{l}}. \quad (6)$$

We use this as the input to both the baseline and proposed models. We observe that this kind of normalization improves the calibration of baseline and proposed models. We do not normalize the probabilities while reporting the ECE of uncalibrated models. We use BIO-tags for training. While decoding, we also allow spans that start with "I-" tag.

A.3.3 QA experiments

We use three datasets for our QA experiments, SQAUD 1.1, EMRQA and MADE 1.0. Our main aim in these experiments is to understand the behaviour of calibration and not the complexity of the tasks themselves. Therefore, we restrict the passage lengths of EMRQA and MADE 1.0 datasets to be similar to SQuAD1.1. We pre-process the passages from EMRQA to remove unannotated answer span instances and reduce the passage length to 20 sentences. EMRQA provides multiple question templates for the same question type (referred to as logical form in Pampari et al. (2018)). For each annotation, we randomly sample 3 question templates for our QA experiments. This is done to ensure that question types that have multiple question templates are not over-represented in the data. For example, the question type for "'Does he take anything for her —problem—" has 49 available answer templates, whereas "How often does the patient take —medication—" only has one. So for each annotation, we sample 3 question templates for a question type. If the question type does not have 3 available templates, we up-sample. For more details please refer to Pampari et al. (2018).

EMRQA is a QA dataset constructed from named entity and relation annotations from clinical i2b2 datasets consisting of adverse event, medication and risk related questions (Pampari et al., 2018). We aim to also test the performance of our calibration method on out-of-domain test data. To do so, we construct a QA dataset from the clinical named entity and relation dataset MADE 1.0, using the questions and the dataset construction procedure followed in EMRQA. This allows us to have two QA datasets with common question forms, but different text distributions. This experimental setup enables us to evaluate how a QA system would perform when deployed on a new text corpus. This corresponds to the application scenario where a fixed set of questions (such as Adverse event questionnaire (Naranjo et al., 1981)) are to be answered for clinical records from different sources. Both EMRQA and MADE 1.0 are constructed from clinical documents. However, the documents themselves have different structure and language due to their different clinical sources, thereby mimicking the real-world application scenarios of clinical QA systems.

MADE QA Construction MADE 1.0 (Jagannatha et al., 2019) is an NER and relation dataset that has similar annotation to "relations" and "medication" i2b2 datasets used in EMRQA. EMRQA uses an automated procedure to construct questions and answers from NER and relation annotations. We replicate the automated QA construction followed by Pampari et al. (2018) on MADE 1.0 dataset to obtain a corresponding QA dataset for the same. For this construction, we use question templates that use annotations that are common in both MADE 1.0 and EMRQA datasets. Examples of common questions are in Table 7. A full list of questions in MADE 1.0 QA is in "question_templates.csv" file included in supplementary materials. The dataset splits for EMRQA and MADE QA are provided in Table 8.

Forecaster features Since we only consider single-span answer predictions, we require a constant number of predictions (answer start and answer end token index), for this task. Therefore we do not use the "ln" feature in this task. The uncalibrated probability of an event is normalized as follows and then used as input to all calibration models.

$$\hat{p} = \frac{1}{M} \sum_{MC-Dropout} p_\theta(y_{start}, y_{end}|x)^{1/2} \quad (7)$$

Unlike the previous tasks, extractive QA with single-span output does not have a varying number of output predictions for each data sample. It always only predicts the start and end spans. Therefore using length normalized (where length is always 2) uncalibrated output does not significantly affect the calibration of baseline models. However, we use the length-normalized uncalibrated probability as our input feature to keep our base set of features consistent throughout the tasks. Additionally, in extractive QA tasks with non-contiguous spans, the number of output predictions can vary and be higher than 2. In such cases, based on our results on POS and NER, the length-normalized probability may prove to be more useful. The "Var" feature and "Rank" feature is estimated as described in previous tasks.

Input	Output	Example Question Form
Problem	Treatment	How does the patient manage her —problem—
Treatment	Problem	Why is the patient on —treatment—
Problem	Problem	Has the patient ever been diagnosed or treated for —problem—
Drug	Drug	Has patient ever been prescribed —medication—

Table 7: Examples of questions that are common in EMRQA and MADE QA datasets.

Dataset Name	Train	Validation	Test
EMRQA	74414	8870	9198
MADE QA	99496	14066	21309

Table 8: Dataset size for the MADE dataset QA pairs that were constructed using guidelines from EMRQA. EMRQA dataset splits are also provided for comparison.